Detecting Changes in Process Behavior Using Comparative Case Clustering

B.F.A. Hompes[1,2], J.C.A.M. Buijs[1], W.M.P. van der Aalst[1], P.M. Dixit[1,2], and J. Buurman[2]

[1] Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Philips Research, Eindhoven, The Netherlands
{b.f.a.hompes,j.c.a.m.buijs,w.m.p.v.d.aalst}@tue.nl
{prabhakar.dixit,hans.buurman}@philips.com

Abstract. Real-life *business processes* are complex and often exhibit a high degree of variability. Additionally, due to changing conditions and circumstances, these processes continuously evolve over time. For example, in the healthcare domain, advances in medicine trigger changes in diagnoses and treatment processes. Case data (e.g. treating physician, patient age) also influence how processes are executed. Existing *process mining* techniques assume processes to be static and therefore are less suited for the analysis of contemporary, flexible business processes. This paper presents a novel *comparative case clustering* approach that is able to expose changes in behavior. Valuable insights can be gained and process improvements can be made by finding those points in time where behavior changed and the reasons why. Evaluation using both synthetic and real-life event data shows our technique can provide these insights.

Keywords: process mining, trace clustering, concept drift, process comparison

1 Introduction

The execution of business processes is typically influenced by many external factors. Due to changing conditions and circumstances, these processes continuously evolve over time. For example, advances in medicine can change how patients are treated or how diagnoses are made in hospitals. Other changing circumstances could be legislation, seasonal effects or even involved resources. As a result, in these flexible processes, many cases follow a unique path through the process. This variability causes problems for existing process mining techniques that assume processes to be structured and in a steady state. Contemporary process mining techniques return spaghetti-like processes and potentially misleading results when this is not the case [3, 9, 18]. The discovered process models capture behavior possible at any given point in time. Often, however, the process context changes and what was possible before is no longer possible or vice-versa. Figure 1 shows the relevance of the problem with an example process that has changed over

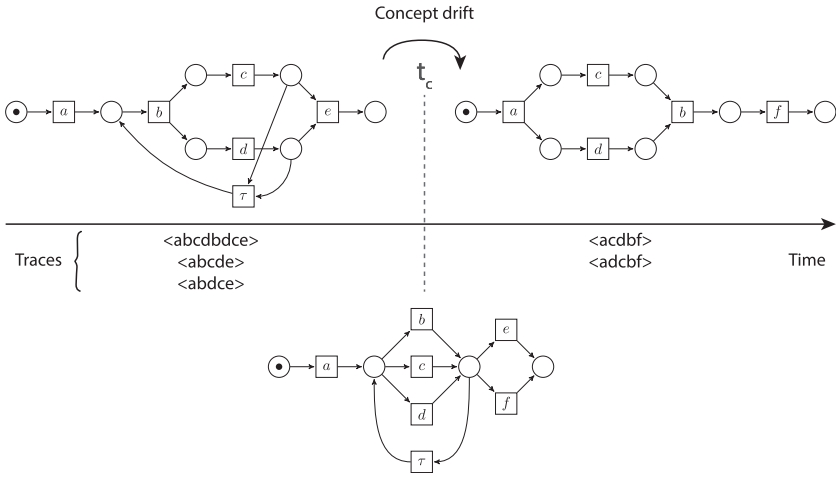

Fig. 1: Demonstration of (sudden) concept drift. The upper process models (Petri nets) accurately describe behavior before and after a change point t_c, whereas the lower model tries to capture all behavior, thereby returning a misleading process model that allows for too much behavior that is not seen in the event log.

time. This so-called *concept drift* is one of the key challenges in process mining, as discussed in the Process Mining Manifesto [2]. Few techniques have been proposed to deal with concept drift in a business process setting [5, 7, 11, 13, 19]. Where existing techniques focus mainly on change in control-flow, our focus is on detecting changing behavior including data aspects.

We identify two types of change for which existing techniques do not work well: individual cases for which behavior changed, and changes in the overall process. The former are typically seen as outliers, because those cases are usually infrequent or dissimilar to the majority of other cases. However, specific changes in context might require cases to change behavior. Alternatively, processes themselves can be subject to change, due to changing conditions and circumstances.

Finding changes in behavior can be of great value to process owners. Detecting unwanted changes, for example, can identify potential risks while positive change can lead to perdurable process improvement. Techniques that consider the entire event log at once cannot show the individual changes that occur throughout the lifetime of a process [5]. For example, specific types of behavior might occur for a limited time only, or can have a seasonal nature. Behavior can also merge with, or emerge from other behavior.

Recently a novel technique for the detection of common and deviating behavior using *trace clustering* was proposed in [10]. Here, the process context is considered by taking both control-flow as well as case and event data into account. Hence, clustering of cases is not limited to finding similar execution paths. In this paper, we extend the work in [10] by providing a novel technique for *change point detection* and a means to compare case clustering results. Figure 2 shows a

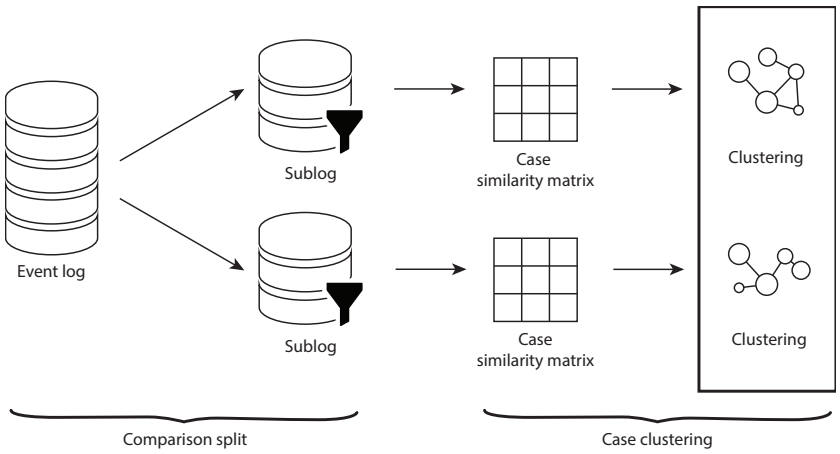

Fig. 2: A graphical overview of the approach. The event log is split into sublogs. Differences in behavior can be compared using comparative case clustering.

high-level overview of our approach. Event logs are split up and the behavior of cases in the resulting sublogs is compared. Our technique aids the analyst by providing indications of the points in time where behavior changed and thus where interesting comparisons can be made. The case clustering technique is based on the Markov cluster (MCL) algorithm [17]. It is able to autonomously discover a (non-predefined) number of clusters of different sizes and densities, leading to the separation of mainstream and deviating behavior.

By utilizing the time dimension we can discover temporal evolutions in process behavior. In order to detect change points, we look at how similarities between cases evolve over time. We consider the effect propagated by new events on a clustering of cases as an indicator of changing behavior. We account for the effect of case maturity in similarity calculation using an aging strategy. For example, it might be the case that certain behavior has not been seen for a while or is seasonal of nature. It would be of interest to analysts to discover the points in time where this behavior has returned. As such, in our technique, similarity between cases is corrected for age. Differences in behavior are then analyzed by comparing clusterings created for two selected partitions of the event logs before and after the detected change point. Partitions include different customer types, cases handled by different resources, etc.

The remainder of this paper is organized as follows. Section 2 briefly introduces necessary preliminary definitions. Section 3 describes the case clustering technique on which our method is built. The detection of change points and related considerations are described in Section 4. Section 5 describes the comparative trace clustering approach. An experimental evaluation on a synthetic and two real-life event logs is performed in Section 6. Section 7 discusses related work. The paper concludes with a summary and planned future work in Section 8.

2 Preliminaries

Typically, the executed *events* of multiple *cases* of a *process* are recorded in an *event log*. An event represents one execution of an activity for a case, and potential contains additional data attributes such as a timestamp or the responsible resource. A trace is a finite sequence of events, and describes one specific instance (i.e. a case) of the process at hand in terms of the executed events. A case can also have additional (case-level) attributes such as a patient birthdate or customer type. Definitions for events and cases are based on those in [1].

Definition 1 (Event, attribute). *Let $\mathcal{E}$ be the event universe, i.e. the set of all possible event identifiers. Events may be characterized by various attributes, e.g. an event may have a timestamp, correspond to an activity, be executued by a particular person, etc. Let N be a set of attribute names. For any event $e \in \mathcal{E}$ and attribute name $n \in N$: $n(e)$ is the value of attribute n for event e. If event e does not have an attribute named n, then $n(e) = \perp$ (null value).*

Typically, the following attributes are present in all events: $activity(e)$ is the *activity* associated to event e, $time(e)$ is the *timestamp* of e, and $resource(e)$ is the *resource* associated to e. Additional event attributes can be the cost associated with the event, the outcome of an activity (e.g. diagnosis result), etc.

Definition 2 (Case, trace, event log). *Let $\mathcal{C}$ be the case universe, i.e. the set of all possible case identifiers. Cases, like events, have attributes. For any case $c \in \mathcal{C}$ and attribute name $n \in N$: $n(c)$ is the value of attribute n for case c ($n(c) = \perp$ if c has no attribute named n). Each case has a mandatory attribute 'trace': $trace(c) \in \mathcal{E}^*$. $\hat{c} = trace(c)$ is a shorthand notation for referring to the trace of a case. A trace is a finite sequence of events $\sigma \in \mathcal{E}^*$, such that each event appears only once, i.e. for $1 \leq i \leq \ j \leq |\sigma|$: $\sigma_i \neq \sigma_j$. For any sequence $s = \langle s_1, s_2, \ldots, s_n \rangle$, $set(s) = \{s_1, s_2, \ldots, s_n\}$ converts a sequence into a set, e.g. $set(\langle a, b, c, b, c, d \rangle) = \{a, b, c, d\}$. An* event log *is a set of cases $L \subseteq \mathcal{C}$ such that each event appears at most once in the entire log, i.e. for any $c, c' \in L$ such that $c \neq c'$: $set(\hat{c}) \cap set(\hat{c}') = \emptyset$.*

In the example cases in Figures 1 and 4 a simplified form is used where events are represented solely by the activities they represent. In this form, a trace is a sequence of activities and an event log a multiset of traces (since in this simplified form cases can share the same sequence of events, and no additional data attributes are present).

In order to automatically separate the event log into multiple sublogs and to detect changes in behavior, we use a time window. Time windows are defined by their length, either in time units or in a specific number of events. This has as an effect that their length in real-time can vary. Behavior in time periods at which less events are recorded (such as low seasons) can thus be represented by a single window, whereas high-frequency periods are divided in more time windows. Here, we use a tumbling time windowing strategy. More detail about the different possible types of time windows is given in Subsection 4.1.

Definition 3 (Time window). *Let W be a time window (window). A window W has two properties: W_s, and W_e, which denote the start and end times of W. The length of W is denoted by $|W| = W_e - W_s$. $L{\restriction}_W = L_W \subseteq L$ denotes the projection of an event log $L \subseteq \mathcal{C}$ to window W. In L_W, case information (attribute values, events) known after time W_e in L is removed, i.e. for all cases, the prefix up to W_e of their trace and case information known at time W_e is kept.*

Cases in an event log can be clustered based on multiple perspectives. Perspectives can be based on case and/or event attributes (such as the age of a patient), simple or more advanced control-flow patterns, or can be derived values such as the time spent in the hospital. By using both control-flow and data perspectives the process context is considered.

Definition 4 (Perspective). *Let P be a perspective. ${\restriction}_P : \mathcal{C} \to \mathbb{R}^m$ denotes the function mapping a case to a real vector of length m according to perspective P. m is the number of attributes in P. For example, m can be the number of different resources in the log or the amount of distinct diagnoses there are. $c{\restriction}_P$ denotes the projection of case $c \in \mathcal{C}$ to a perspective P. Furthermore, we let $c{\restriction}_{\{P_1,P_2,\ldots,P_K\}} = c{\restriction}_{P_1} \| c{\restriction}_{P_2} \| \ldots \| c{\restriction}_{P_K}$, i.e. the resulting profile vectors from projection to multiple perspectives are concatenated.*

The Markov cluster algorithm uses a similarity matrix between cases as its input. This matrix holds pair-wise similarity values between the profile vectors obtained by projecting each case to the selected perspectives.

Definition 5 (Case similarity matrix). *Let $L \subseteq \mathcal{C}$ be an event log.*
$\mathcal{M}(L) = (L \times L) \to [0.0, 1.0]$ denotes the set of all possible case similarity matrices over L. For cases $c, c' \in L$ and a case similarity matrix $M \in \mathcal{M}(L)$, $M(c, c')$ denotes the similarity between c and c'.

Similarity values for cases are multiplied by an age factor that decreases with time in order to correct for the similarity between current and older cases in the change detection process. This way, we account for seasonal temporal changes, or temporally infrequent behavior.

Definition 6 (Age vector). *Let $L \subseteq \mathcal{C}$ be an event log.*
$\vec{a}(L, W) = L \to [0.0, 1.0]$ denotes an age vector a over L, for time window W. For any case $c \in L$, $\vec{a}(c, W)$ denotes the age factor of c for window W.

Definition 7 (Case clustering). *Let $L \subseteq \mathcal{C}$ be an event log. A case cluster (cluster) over L is a subset of L. A case clustering $TC \subseteq \mathcal{P}(L)$[1] is a set of case clusters over L. We assume every case to be part of at least one cluster, i.e. $\bigcup TC = L$. Cases can be in multiple clusters, i.e. cluster overlap is allowed.*

[1] $\mathcal{P}(L)$ denotes the powerset over event log L, i.e. all possible sublogs of L.

3 Case Clustering

Discovering a process model on an entire real-life event log will often lead to a spaghetti model since it has to represent all past traces [3, 9, 18]. Similarly, clustering the entire event log will show groups of behavior that were possible at any given point in time. The temporal evolution of the behavioral differences captured by the clustering is not shown.

Our change detection technique is based on the technique proposed in [10] where case clustering and outlier detection are combined in order to find mainstream and deviating behavior. This technique relies on the Markov cluster (MCL) algorithm [17] to find clusters of cases that share behavior on a set of selected perspectives. By incorporating both control-flow and case data, the process context is taken into account. MCL was chosen over alternative clustering techniques because of the following properties. The number of clusters is discovered rather than set beforehand, hence changes in behavior will be reflected in change in the clustering. Because MCL is neither biased towards globular or local clusters and is able to find clusters of different density, exceptional cases will not be clustered together with common behavior, i.e. they can be distinguished based on cluster sizes. As a result, new types of cases will result in new clusters rather than to be added to existing clusters. Since cluster overlap is possible, evolution of cases from one cluster to another over time can be detected as well. For more details on differences with alternative clustering approaches the reader is referred to [10].

MCL uses a stochastic similarity matrix between cases as its input. It alternates an expansion step that raises the matrix to a given power with the inflation step. Inflation raises each element to a given power and normalizes the matrix such that it is stochastic again. As such, there are two parameters to MCL: the expansion and inflation parameter, which both influence clustering granularity. This alternation eventually results in the separation of the matrix into different components, which are interpreted as clusters. In our case, the MCL algorithm takes as input a left stochastic version of the case similarity matrix, i.e. the columns are normalized. In order to create a case similarity matrix, cases in the event log are mapped to a profile vector by projecting them to a selected set of perspectives. A pair-wise similarity score is then calculated between these profile vectors. Applying the MCL algorithm to the resulting case similarity matrix gives us a case clustering over the log. The process of applying MCL to case clustering is visualized in Figure 3.

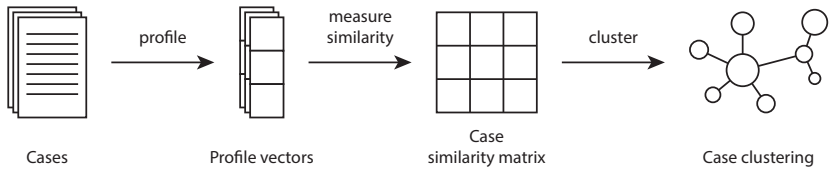

Fig. 3: Overview of the application of the Markov cluster algorithm.

More formally, for all cases $c \in L \subseteq \mathcal{C}$, we project c to our chosen set of perspectives P, i.e. $c{\restriction_P}$. Next, for each pair of the resulting profile vectors, we compute the pair-wise vector similarity. In this paper we use the cosine similarity, i.e. $M(c, c') = \frac{c{\restriction_P} \cdot c'{\restriction_P}}{||c{\restriction_P}|| \; ||c'{\restriction_P}||}$, $\forall c, c' \in L$, where $M \in \mathcal{M}(L)$. However, any vector similarity metric can be used. Cosine similarity was chosen because of its proven effectiveness, efficient calculation and boundedness to $[0, 1]$. Also, it is able to represent non-binary term weights and allows for partial matching. A typical downside of vector similarity measures is that the order of terms is lost. This problem can be solved by incorporating order in the perspectives, such as the occurrence of frequent patterns.

4 Detecting Change in Behavior

In the extreme case, we could cluster the cases in the event log after every new event that arrives, with the aim of finding significant changes in behavior. However, due to the nature of different clustering algorithms (including the Markov cluster algorithm), this is too time and resource consuming. Often, even clustering after a specific number of new events or time units would be too expensive still, or would need excessively large window sizes to be feasible.

The Markov cluster algorithm autonomously discovers a number of clusters with varying sizes and densities based on its input. Over time, the occurrence of new events will change the similarity between cases, which is reflected in the similarity matrix. As this matrix is the input for the clustering algorithm, the impact on the case similarity matrix is a good indicator for how much the clustering will change. Hence, we can use the evolution of this matrix over time as a reliable predictor for change in the clustering output. Therefore, we propose to detect changes in behavior by utilizing the change over time of the case similarity matrix. Similar to the approaches that use statistical tests [5, 12, 13], differences in the matrix indicate change in behavior. An overview of our approach is illustrated in Figure 4, where five simple example traces are drawn over time.

4.1 Splitting the event log

In order to calculate change in behavior, the events belonging to the different cases are split over several consecutive time windows. Different windowing strategies exist in literature, such as adaptive, tumbling, sliding, and flexible windows. In our case, we use a tumbling windowing technique that facilitates the comparison of cases from their start to the end of the window. For each time window, cases that have events in or before that window are considered in the calculation of the similarity matrix, as is depicted in Figure 5. Events occurring after the current time window are not considered. As such, for each case, only the known attributes and prefix of its trace are taken into account. Different window sizes can be considered. For example, we can compute a new similarity matrix every 10 events, or after every 30 minutes. The choice for sensible window sizes depends entirely on the type of event data that is being analyzed. As such, no rule of thumb can be provided.

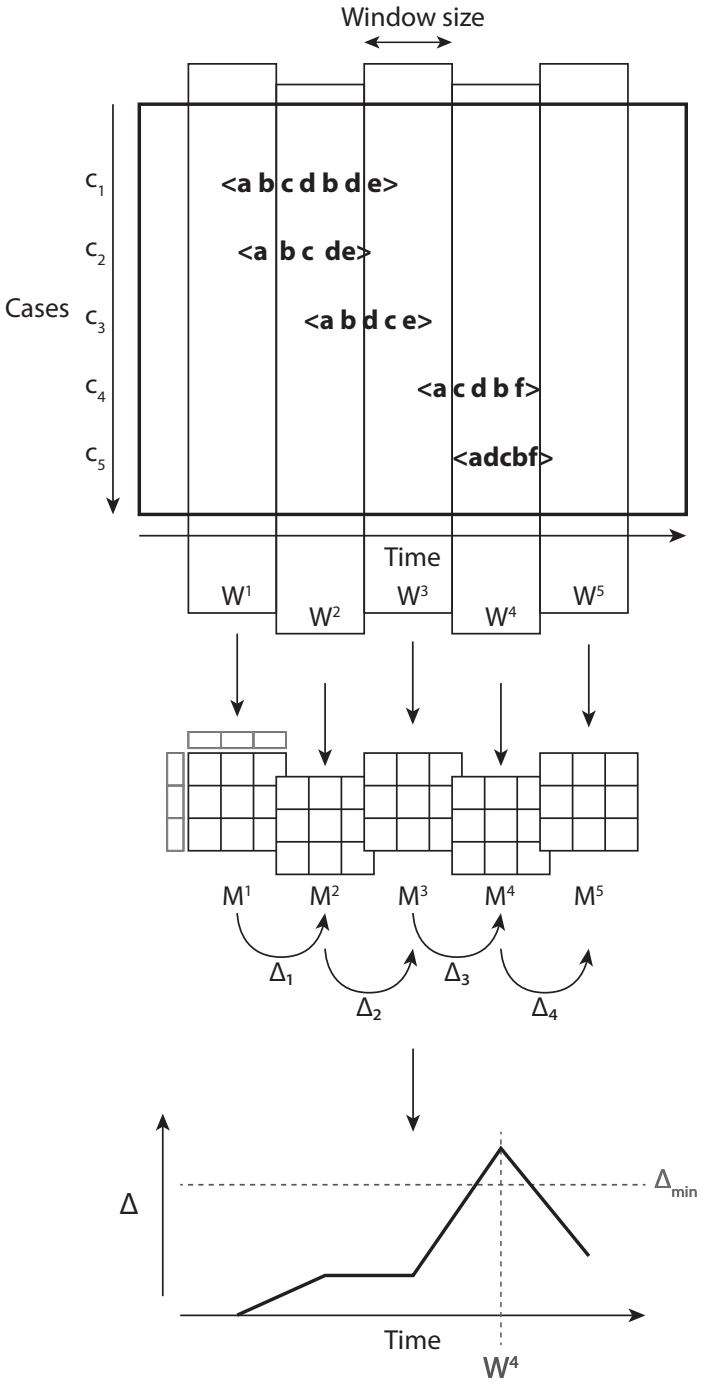

Fig. 4: A graphical overview of the change detection approach. In order to detect changes in behavior, the difference in similarity matrices over time is calculated. In this example, significant change occurred in window W^4.

Note that it is not possible to use windowing techniques that vary the window size based on the automated detection of changes in the data, since this detection is the goal of our approach. We consider attributes that are not specifically output by events (such as patient age) to be known from the very first event in the trace of a case. As a result, when looking at this type of case-level data, a change point will be shown for the first event of a case. Related research has been performed in the field of streaming data [8], and process mining in a streaming setting in particular [6].

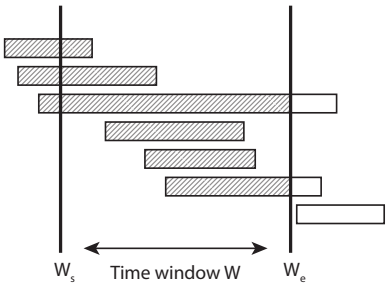

Fig. 5: Selection of cases using a time window. Cases are represented by horizontal bars. All events for all cases up to W_e are taken into account.

Formally, for every time window W^i, where $i \in [1, n]$, an event log $L \subseteq \mathcal{C}$ is projected to W^i, resulting in n sublogs $L{\restriction}_{W^i}$. Note that $W^i_s = W^{i-1}_e$, for all $1 < i \leq n$, i.e. all windows are consecutive, and there is no overlap in their start and end times. For every W^i, $L{\restriction}_{W^i}$ contains all events in L up to time W^i_e. Also note that as discussed, $|W^i|$ is not necessarily equal to $|W^{i+1}|$, for example when we choose a number of events as the size of the time windows.

4.2 Calculating case similarity

After splitting the events in the event log over consecutive time windows, a similarity matrix is calculated for each sublog and compared with the similarity matrix calculated for the sublog that was obtained by projecting the event log to the previous time window. Before this is done, however, we account for case age.

Without correcting for the age of cases, the effect of seasonal temporal behavior patterns are hidden. For example, imagine a process that is executed differently in winter (low season) compared to the other three season. Without accounting for case age, cases that are observed in winter will exhibit a high similarity with those cases observed in winter the previous year. No change will be detected in the transition from fall to winter. Similarly, cases that are exceptional but not unique will become less detectable over time.

In order to correct for the age of cases, we keep a vector of case age factors, that are used to decrease case similarity for those cases further back in history. This factor can be reduced over time in different ways (e.g. exponentially or linearly), in order to consider only recent cases or also the earlier ones. So, for every sublog $L{\restriction}_{W^i}$, we create a case similarity matrix M^i, as described in Section 3. This matrix is corrected for age by multiplying it by $A^i = \sum_{j=1}^{|L{\restriction}_{W^i}|} \vec{a}(L, W^i)_j \cdot E_j$, where E_j is the $|L{\restriction}_{W^i}| \times |L{\restriction}_{W^i}|$ matrix with a 1 on position (j, j) and zeros everywhere else. In the resulting case similarity matrix only the upper triangle has been accounted for age. We mirror the upper triangle downwards to make the

matrix symmetrical again. To exponentially decrease the impact of a case to the case similarity matrix, we multiply the age vector with some aging factor in every new time window, i.e. $\vec{a}(L, W^i) = \vec{a}(L, W^{i-1}) \cdot s$, where $s \in [0, 1]$. Alternatively, in order to decrease the age linearly, we can subtract a value for each element in the vector instead of multiplying with a factor.

Note that this type of correction for age combined with the windowing technique described in Subsection 4.1 differs in effect from traditional sliding window techniques, since similarity between cases might result from attributes or events that were known before a certain time window. As such, we decrease the impact of the similarity value obtained by comparing all case data known so far, rather than computing similarity only on a subset of that data.

4.3 Computing change

Once age has been accounted for, the difference in similarity matrices can be calculated. For the calculation of the change in similarity matrices, we look at the change in the matrix values. The maximal difference equals the amount of cells we have in the case similarity matrix of the latest time window divided by two (since the matrix is symmetrical). Hence, we calculate the change value as a percentage of this amount. Formally, the change between two case similarity matrices M^1 and M^2 of size k equals $\frac{2}{k} \times \sum(|M^2 - M^1|)$, where $\sum M = \sum_{c,c'} M(c, c')$. Note that the dimensionality of similarity matrices changes over time due to the cases that are included in the respective time windows. When two matrices of different dimensions are compared, both matrices are extended with empty cells referring to cases that are only present in the other time window.

Interesting change points can be deduced from the evolution of the change value over time. An increase indicates that in this window, more things changed compared to in the previous window. A drop indicates less change. Accordingly, interesting change points are those points in time where the change value exhibits spikes. Once changes in behavior have been discovered, clusterings can be created in order to compare behavior before and after the identified change points.

For example, take the cases depicted in Figure 4, and consider cases to be similar when they share activities. At first, up to event window W^3, case c_4 seems to be quite similar to the other three cases seen so far. However, in window W^4 this changes due to activity f which has replaced activity e. This change is reflected in the similarity matrices M^1 to M^5. Consequently the value for Δ_3 indicates a possible change point in W^4.

5 Comparative Clustering

The previous two sections describe how to identify change in process behavior based on the change in similarity between cases. Once interesting change points have been identified, different clusterings can be created and compared, both programmatically and visually, in order to analyze the effect of changing behavior in a process. By comparing clusterings created for different sublogs (i.e. before

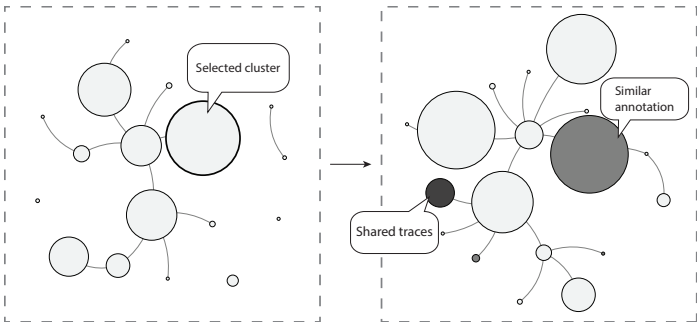

Fig. 6: Two clusterings are compared. Highlighted clusters on the right clustering show how behavior captured by the cluster selected on the left has changed.

and after an identified change point), we can see where behavior changed and analyze why. Of course, it is also possible to manually create selections of cases based on time or data attributes in order to compare behavior. This can, for example, be used to compare behavior for different age groups, for patients from different geographical locations, different years, etc.

Due to the properties of the MCL algorithm discussed in Section 3 and Section 4, changing behavior is reflected in changes in the cluster structure. For example, cluster sizes indicate the frequency of the captured behavior. Behavior that used to be common but is becoming less frequent will still be clustered separately rather than get merged with different or common behavior, as long as the behavior remains dissimilar.

Clusters of cases are represented as nodes in a partially connected, undirected graph. Node sizes indicate the number of clustered cases. Two nodes are connected when there is a positive similarity between at least one pair of cases between the two represented clusters. In more advanced visualizations, these edges could be given a weight to represent minimum, maximum, or average similarity, etc. This can be used to visually show inter-cluster similarity. Also, edge weights can be used in layout calculation for complex cluster graphs. Clusters are annotated with descriptions about the cases that are present. Shared activities between traces and similar data values are a few of the possibilities. For example, a cluster that groups cases of patients that all share a certain diagnosis can be annotated with that diagnosis description.

In Figure 6, two example clusterings are compared. In the left clustering, one cluster is selected. By highlighting those clusters in the right clustering that share behavior with the selected cluster(s) on the left, we can interactively analyze how behavior has changed. It is possible to see how cases are clustered or what annotations are shared. For example, we might compare two sublogs of patient data, before and after an identified change in behavior. Patients that are present in both years are highlighted as shared cases (in dark gray), while clusters that share some or all diagnoses (similar annotations) are highlighted (in light gray). Within a case clustering, clusters that share behavior are connected. As such,

when comparing two clusterings, clusters that are split into or have emerged from multiple clusters can be found by looking at the clusters sharing annotations and/or cases. This can, for example, indicate that behavior has become more specific or more general. Besides visual approaches, techniques such as [20] that aim to explain clusters of cases can be used as well.

6 Evaluation

In order to evaluate our change detection technique we use a synthetic event log and two publicly available real-life event logs. In Subsection 6.1 we show how change in behavior over time can be detected using our approach. In Subsection 6.2 we apply our technique to uncover useful and interesting insights from real-life data. Our technique has been implemented in the process mining tool ProM[1], and is publicly available through the *TraceClustering* package.

6.1 Synthetic evaluation

For the synthetic evaluation, an event log was generated from a fictive, manually created radiology process with 17 unique activities. 1,000 cases were generated spanning one year. The control-flow of this process heavily depends on the data-attributes 'age' and 'bodypart', as well as the time of year the patient arrives. As a result, there are many possible control-flow variants recorded in the event log. We distinguish four different scenarios, based on patient age and the part of the body the radiology exam is to be made of. At the same time, the process has been constructed in such a way that temporal patterns in behavior occur, as the inflow of types of patients is seasonal. Special patients (patients younger than 10 years old that need a head exam) were modeled to arrive only in the months March and April, July and August, and in November and December. Patients of other ages and types can arrive all year long. In other words, the behavior of cases changes over time.

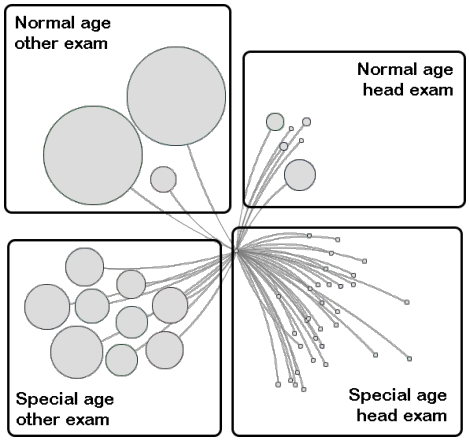

Fig. 7: Changes in behavior over time are hidden when clustering the entire synthetic event log and disregarding time.

Figure 7 shows an example result of clustering the entire event log on the occurrence of activities, without regarding the time aspect. Clustering all cases in the log shows all behavior

[1] See `http://promtools.org`

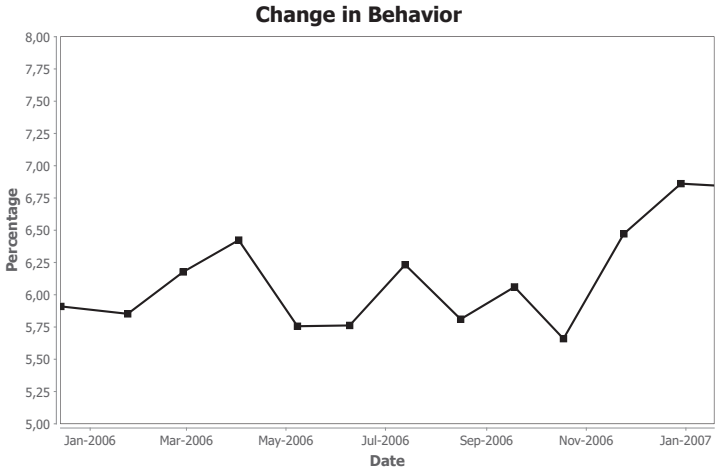

Fig. 8: Change in behavior over time for the synthetic event log. Changes in behavior can be identified by comparing case similarity matrices for consecutive time windows.

that was possible at any given point in time. As a result, we can only identify the different groups of patients, no changes in behavior or seasonal pattern are discovered.

By applying our change detection technique we can find changes in behavior over time. A new similarity matrix was created for the event log and compared with the previous matrix, every 100 events. Note that the choice of window size affects the perceived change, as larger window sizes smooth out local changes in behavior. Figure 8 shows the calculated change over time. The horizontal axis represents time whereas the vertical axis indicates the percentage of change occurring in each time window compared to the previous window. The change values indicate a change in behavior when the first special patients started to arrive in March. In Figure 8, the age of cases is accounted for, according to the technique described in Subsection 4.2. For every time window, the age factor of each case was multiplied by 0.95 in order to make sure that ages of older cases were accounted for. We can see that new change points are discovered in July and November, indicating a (potential) seasonal pattern, and interesting points in time to create new clusterings upon. As is expected, the first event windows indicate big change values since all information seen in them is new.

6.2 Real-life data

The first real-life log comes from a Dutch academic hospital that contains cases pertaining to cancer treatment procedures. It was originally used in the first Business Process Intelligence Contest (BPIC 2011) [15]. The second event log contains cases of building permit applications provided by a Dutch municipality.

This log is part of the 2015 edition of the BPI Challenge (BPIC 2015) [16]. These event logs were used so that the results can be reproduced. In the results shown here, we used the following MCL parameters: expansion = 2, inflation = 15.

The first event log contains cases of different stages of malignancy and of different parts of the body. Also, information is present about the diagnosis, treatment, specialism required, patient age, organisational group (hospital department), etc. This log contains 1,143 cases, 150,291 events and 624 distinct activities. There are 981 different executions paths (activity sequences). There are many attributes present on both the event and case level. All of these attributes can obtain several different values, leading to a large heterogeneity in the log. As cases are recorded between January 2005 and March 2008, the event log is likely to exhibit drifts in control-flow and changes in process behavior.

For each case in the hospital log, there are 16 attributes for 'diagnosis code', referring to the diagnoses the patient received for different parts of their body. By comparing on these attributes and calculating the change in behavior over time, we found that near July 2006, a change in diagnoses occurred. The change in behavior was calculated every 5,000 events, and every window the age factor of cases was multiplied by 0.95.

Figure 9 shows how the behavior has evolved. In Figure 10, the clustering on the left represents cases two months before the change point whereas the clustering on the right represents cases two months after. Patients that were in the selected cluster and have had activities in both years are highlighted in dark gray. Groups of patients that have had (partially) shared diagnoses are marked light gray. We can see that some diagnoses are present for more body parts and now occur in other combinations. This could indicate a trend in diseases or be due to an improvement in diagnosis detail. As there are many smaller clusters in July-August that have additional diagnoses (light gray), we can deduce that for the selected diagnosis, the related diagnoses have become more specific and diagnoses are also made on other parts of the body. In Figure 10 process maps and differences in activities are shown for two highlighted clusters.

Besides diagnosis codes, every case has 16 possible attributes for 'treatment code', referring to the treatments the patient received on different parts of their body. This leads to many possible treatment combinations for different diagnoses. We inspect the change in behavior over time for the year 2006, when looking at diagnosis code and treatment code. As can be seen in Figure 11, using our technique, a potential change point can be found in November 2006. By comparing the clustering results for September-October and November-December, changes in treatments for specific diagnoses become visible. As we can see from Figure 12, treatment for some diagnoses have changed. Again cases active in the two months before the change point are shown on the left and cases active in the two months after are shown on the right. We can see that there are two clusters that contain patients that were active in September-October, one of which is much smaller than the other. For the larger cluster, additional diagnoses were made and additional treatments were performed. As a result, more cases share this behavior. The call-outs in Figure 12 again show differences in activities between the two periods.

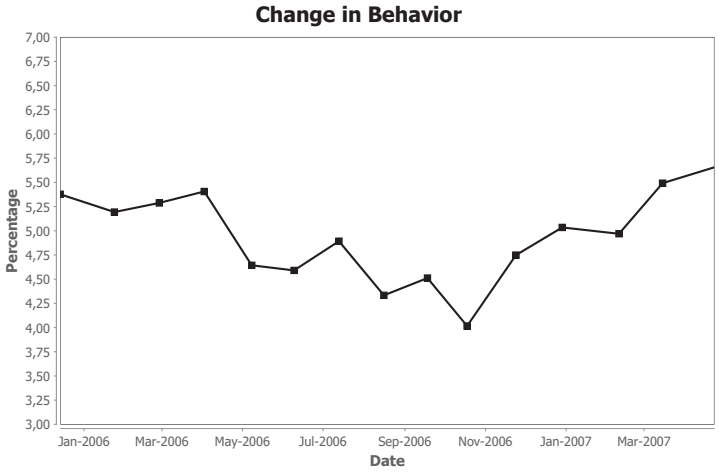

Fig. 9: Change in behavior over time for the hospital log, for the year 2006 and early 2007. Cases are compared on diagnosis code. Potential change in behavior is indicated in July 2006.

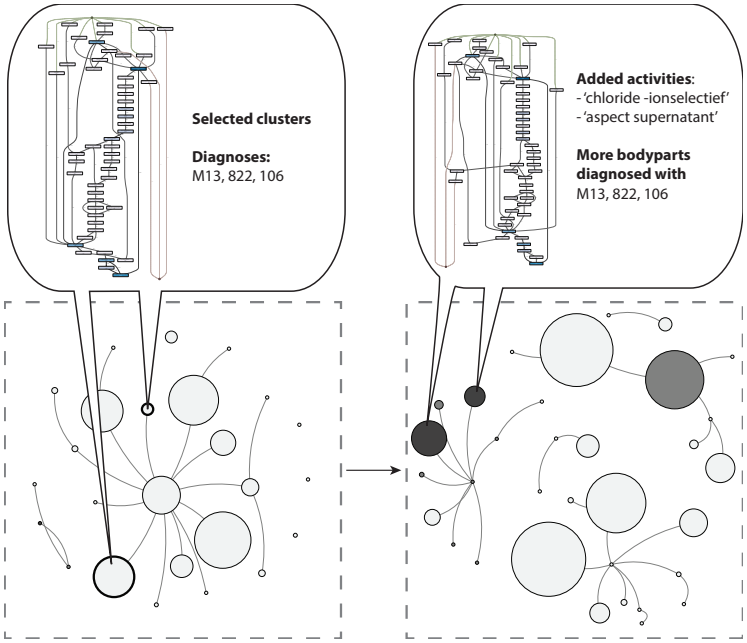

Fig. 10: Hospital log clustered on diagnosis code for cases active in May-June 2006 (left) and July-August 2006 (right). Changes in diagnoses are discovered. More bodyparts are diagnosed with codes M13, 822 and 106.

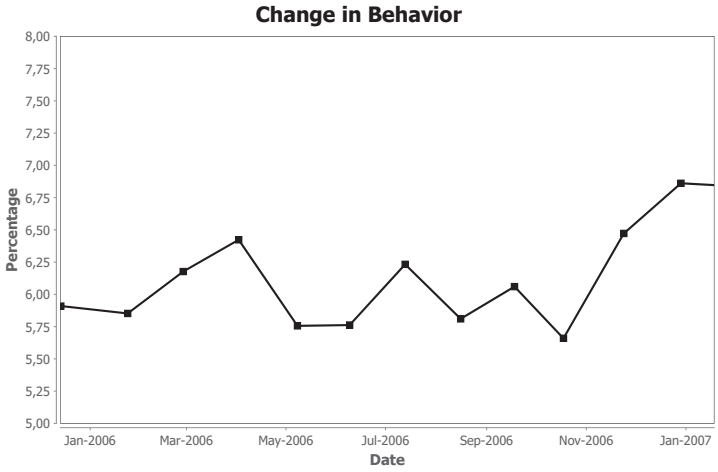

Fig. 11: Change in behavior over time for the hospital log compared on diagnosis code and treatment code, for the year 2006. Potential change in behavior is indicated in November 2006.

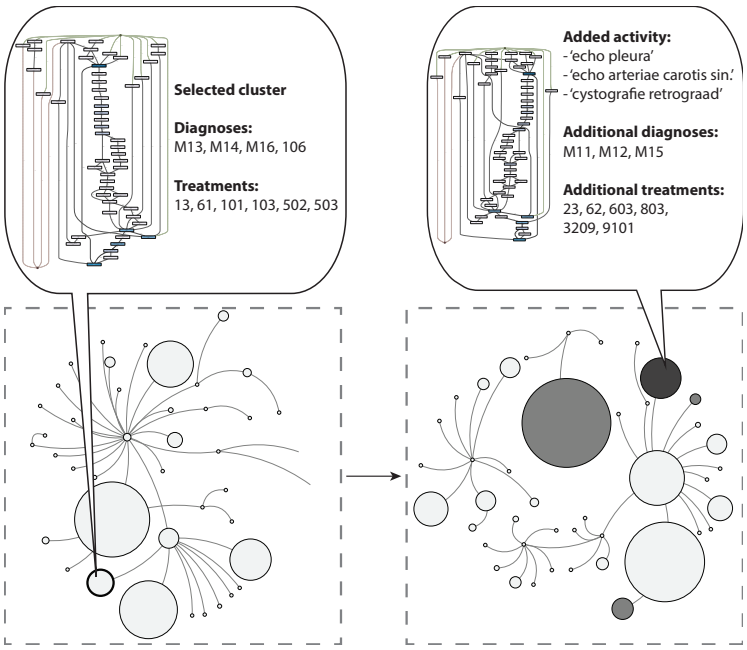

Fig. 12: Hospital log clustered on diagnosis code and treatment code for cases active in September-October 2006 (left) and November-December 2006 (right). Additional diagnoses and treatments are found.

These differences indicate that, over time, treatments for certain diagnoses have changed. Considering the type of process, this could be due to specific patient needs, changes in protocols or advances in medicine. A probable reason is that diagnoses were made (or recorded) with greater detail. Insights such as these can be gained easily and can be used to verify or specify protocols, check whether certain behavior is changing or for auditing purposes.

The second real-life event log contains cases of building permit applications in a Dutch municipality. Information is present about the type of permit, the costs associated with the permit, the involved resources, etc. Again, each attribute can have several different values. This log contains 1,199 cases recorded between late 2010 and early 2015 with in total 52,217 events and 398 distinct activities. As there are 1,170 different execution paths, almost all cases are unique from the control-flow perspective.

Each case in the municipality log has an attribute 'parts' that refers to the different permit types that are involved in the case it describes. Each case is also labeled with the attribute 'term name', describing which status has been assigned to the permit application. Possible values are 'permit granted', 'additional information required', 'term objection and appeal', etc. Figure 13 shows the change in behavior over time when comparing cases on these two attributes, for the years 2012-2013. The change in behavior was calculated every 2,500 events, and in every window the age factor of cases was multiplied by 0.95. A potential change point is indicated near mid January 2013. We cluster the cases in the log on both permit type and term name and compare cases in December 2012-January 2013 with cases in February-March 2013. The results are shown in Figure 14. As we can see, few clusters are discovered, indicating only slight differences in behavior on these perspectives. A group of cases pertaining to mainly construction and environmental permits that are in the 'objection and appeal' term is selected in the left clustering. During the selected period, most of this behavior has merged with the biggest group of cases, which now represents almost all behavior in the log in the right clustering.

6.3 Effect of parameters

As explained, several parameters are important for obtaining the points in time where behavior has changed. Firstly, the perspectives used to create a case similarity matrix decide on which perspective change is detected. It is therefore important to choose those perspectives that are of interest to the analysis.

The MCL clustering technique uses two parameters, expansion and inflation, which both affect clustering granularity. When change on a low level is of interest, expansion can be decreased and inflation increased, and vice-versa for when only high-level change points are required. Besides the MCL parameters, the window size can also be adjusted to affect the detection span of the approach. Bigger window sizes will result in more global, high-level changes being detected while small windows will also reveal smaller changes in behavior.

The effect of long-running cases and seasonal behavior can be controlled by adjusting the age factor. Increasing the age factor (to a value close to 1) will lead

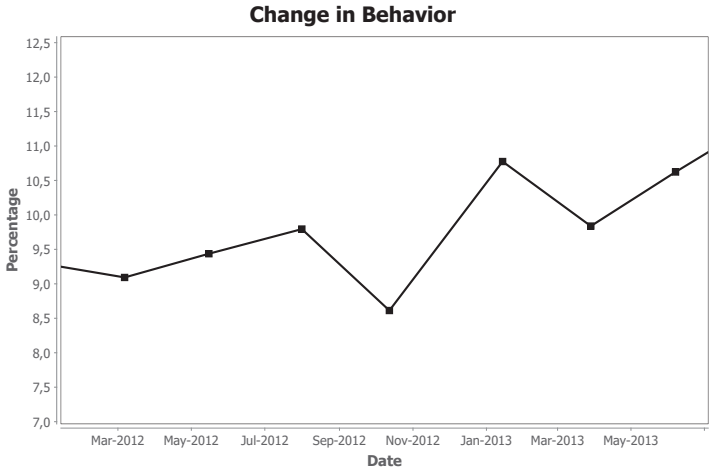

Fig. 13: Change in behavior over time for the municipality log compared on permit type and term description, for 2012-2013. Potential change in behavior is indicated in January 2013.

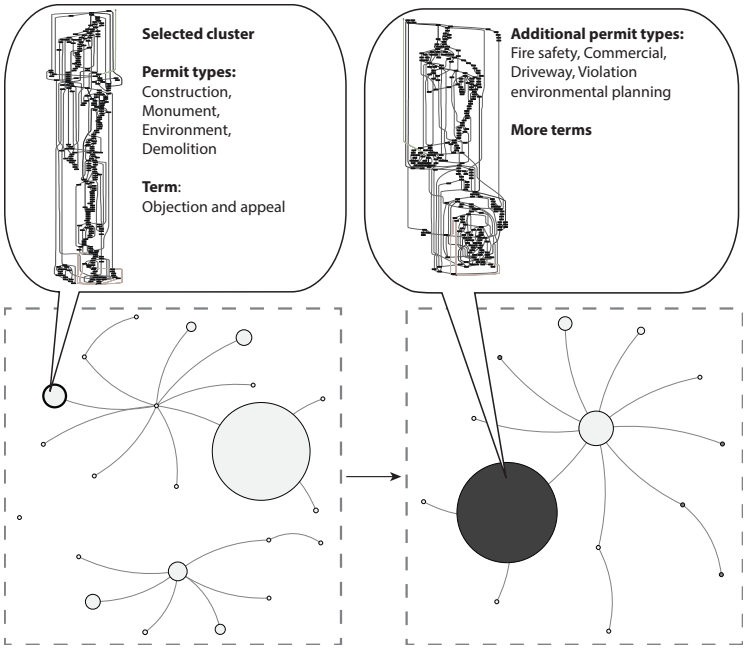

Fig. 14: Municipality log clustered on permit type and term description for cases active in December 2012-January 2013 (left) and February-March 2013 (right).

to longer lasting effects whereas decreasing the age factor will also show seasonal temporal behavior. In conclusion, the setting of parameter values needs to be decided on a case-per-case basis. Most interesting insights will be gained when the approach is used in an iterative process.

7 Related Work

Although concept drift is a well-studied topic in the data mining and machine learning communities, little work has been done on detecting concept drift in business processes. Bose et al. were the first to consider concept drift and change detection in a process mining setting [5]. In their work, a classification of possible changes in business processes is given, and statistical hypothesis tests are used to detect regions of change. Even tough the authors consider the possibility of change in data attributes, the scope of their work is limited to the detection of control-flow changes in a process manifested as sudden drifts over a period of time. More recently, Martjushev et al. built on this work by looking at gradual and multi-order dynamics to detect concept drift in control-flow [13]. They extend the work in [5] by providing solutions to detect gradual change as well. By considering multi-order dynamics through the use of an adaptive window technique, process change occurring at multiple levels of mixed time granularity can be detected. Maaradji et al. employ statistical tests over the distributions of runs observed in two consecutive time windows in order to detect concept drift [12]. As noted by the authors, in order to find differences in process behavior a notion of equivalence is necessary. In their paper, a notion of run-equivalence is used. It is shown that drift can be identified fast and accurately by using an adaptive sliding window technique. As a result, it can be used in an online (streaming) setting as an oracle as to when a discovered model should be updated.

Weber et al. employ probabilistic deterministic finite automata (PDFA) to represent the probability distributions generated by process models [19]. Similar to [5] and [12], statistical hypothesis tests are used to detect whether or not a distribution has changed significantly from a ground truth. The aim of their technique is to identify process change as soon as possible, but with confidence that change is significant, in order to discover a model representing reality as good as possible. As such, only drift in control-flow is considered. In [7] a different technique is proposed to automatically detect and manage concept drift in an online setting. Here, concept drift is detected real-time using an estimation technique based on abstract interpretation of the process and sequential sampling of the log. The fitness of prefixes of new samples taken from the log is checked against that of prefixes of initial samples. A change point is identified when there is a significant difference between these two points. In the above-mentioned techniques however, data attributes are not considered. As such, only changes in control-flow behavior can be discovered. Moreover, case maturity is not accounted for, leading to issues in discovering seasonal temporal changes in behavior.

Trace clustering techniques are often used to find different process variants. Several trace clustering techniques have been proposed in the field of process

mining, and an extensive comparative analysis of trace clustering techniques has recently been performed in [14]. Often, however, the temporal dimension is not considered. In [11], the starting time of each process instance is used as an additional feature in trace clustering. By combining control-flow and time features, the clusters formed share both a structural similarity and a temporal proximity. The technique is based on the technique proposed in [4] and considers different types of changes, including sudden, recurring, gradual, and incremental changes. In more complex evolving business processes however, including the temporal proximity of cases might lead to misleading results. For example when seasonal drifts are intertwined with gradual changes in the process.

The technique proposed in this paper uses similar ideas and concepts as used in the papers mentioned above. However, trace clustering techniques and concepts are used to find changes in common and deviating process behavior. By taking into account both the control-flow and the data aspects, the technique is made context-aware. We extend the technique in [10] by including change detection in behavioral similarities between cases. The input is limited to the perspectives on which we want to cluster and compare behavior, and the two numerical parameters for the Markov cluster algorithm. It is not necessary to manually select the number of desired clusters, as that is determined by the underlying cluster algorithm, along with the cluster sizes and densities. Additionally, different windowing strategies, sizes and aging factors can be used to find different types of drift.

8 Conclusions and Future Work

Real-life *business processes* are often complex while exhibiting a high degree of variability. Due to changing conditions and circumstances, these processes continuously evolve over time. Existing *process mining* techniques assume the process to be static and are less suited for the analysis of contemporary business processes. In this paper we presented a novel *comparative case clustering* approach that is able to expose temporal changes in behavior in a process. By using both control-flow and case data we take the process context into account. Insights can be gained into how and why behavior has changed by comparing changes in clusterings over different partitions of the log. Interesting points in time can be discovered as to give an idea on where to partition the event log. The discovered information can then be used for further analysis, e.g. to design protocols, for early detection of unwanted behavior or for auditing purposes. Besides the time dimension, different data and control-flow attributes can be utilized in order to distinguish groups of behavior.

Our results show that indeed promising insights can be achieved. Nonetheless there are drawbacks. It is necessary to manually select the perspectives on which case similarity is calculated and what window size is used. Also, once change points have been identified, the parameters for the Markov cluster algorithm need to be chosen. Besides the parameters, at the moment, it is not possible to distinguish between changing behavior localized to a specific cluster and

more global change. Additional research is needed to further automate the analysis process, for example by automatically detecting discriminating clustering perspectives or by suggesting parameters for the clustering algorithm. In the future we would also like to look into how changes in process behavior can be analyzed in an online, streaming event data, setting. Different ways to visualize change in behavior can be explored as well.

Bibliography

[1] W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, Berlin, 2011.

[2] W.M.P. van der Aalst, A. Adriansyah, A.K.A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, R.P.J.C. Bose, P. van den Brand, R. Brandtjen, J.C.A.M. Buijs, et al. Process Mining Manifesto. In *Business Process Management Workshops*, pages 169–194. Springer, 2012.

[3] R.P.J.C. Bose and W.M.P. van der Aalst. Context Aware Trace Clustering: Towards Improving Process Mining Results. In *Proceedings of the SIAM International Conference on Data Mining*, pages 401–412. Society for Industrial and Applied Mathematics, 2009.

[4] R.P.J.C. Bose and W.M.P. van der Aalst. Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models. In *Business Process Management Workshops*, pages 170–181. Springer, 2010.

[5] R.P.J.C. Bose, W.M.P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy. Handling Concept Drift in Process Mining. In *Advanced Information Systems Engineering*, pages 391–405. Springer, 2011.

[6] A. Burattin, M. Cimitile, F.M. Maggi, and A. Sperduti. Online Discovery of Declarative Process Models from Event Streams. *IEEE Transactions on Services Computing*, 8(6):833–846, 2015.

[7] J. Carmona and R. Gavalda. Online Techniques for Dealing with Concept Drift in Process Mining. In *Advances in Intelligent Data Analysis XI*, pages 90–102. Springer, 2012.

[8] Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

[9] S. Goedertier, J. De Weerdt, D. Martens, J. Vanthienen, and B. Baesens. Process Discovery in Event Logs: An Application in the Telecom Industry. *Applied Soft Computing*, 11(2):1697–1710, 2011.

[10] B.F.A. Hompes, J.C.A.M. Buijs, W.M.P. van der Aalst, P.M. Dixit, and J. Buurman. Discovering Deviating Cases and Process Variants Using Trace Clustering. In *Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC), November 5-6, Hasselt, Belgium*, 11 2015.

[11] D. Luengo and M. Sepúlveda. Applying Clustering in Process Mining to Find Different Versions of a Business Process that Changes over Time. In *Business Process Management Workshops*, pages 153–158. Springer, 2012.

[12] A. Maaradji, M. Dumas, M. La Rosa, and A. Ostovar. Fast and Accurate Business Process Drift Detection. In *Business Process Management*, pages 406–422. Springer, 2015.

[13] J. Martjushev, R.P.J.C. Bose, and W.M.P. van der Aalst. Change Point Detection and Dealing with Gradual and Multi-order Dynamics in Process Mining. In *Perspectives in Business Informatics Research*, pages 161–178. Springer, 2015.

[14] T. Thaler, S.F. Ternis, P. Fettke, and P. Loos. A Comparative Analysis of Process Instance Cluster Techniques. In *Proceedings of the 12th International Conference on Wirtschaftsinformatik. Internationale Tagung Wirtschaftsinformatik (WI-15), March 3-5, Osnabrck, Germany*. Universitt Osnabrck, 3 2015.

[15] B.F. van Dongen. Real-life Event Logs - Hospital Log, 2011. `doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54`.

[16] B.F. van Dongen. BPI Challenge 2015, 2015. `doi:10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1`.

[17] S. Van Dongen. A Cluster Algorithm for Graphs. Technical report, National Research Institute for Mathematics and Computer Science in the Netherlands, 2000.

[18] G.M. Veiga and D.R. Ferreira. Understanding Spaghetti Models with Sequence Clustering for ProM. In *Business Process Management Workshops*, pages 92–103. Springer, 2010.

[19] P. Weber, B. Bordbar, and P. Tino. Real-Time Detection of Process Change using Process Mining. In *Imperial College Computing Student Workshop*, pages 108–114, 2011.

[20] J. De Weerdt and S.K.L.M. vanden Broucke. SECPI: Searching for Explanations for Clustered Process Instances. In *Business Process Management*, pages 408–415. Springer, 2014.